Defending Against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems

Shucheng Yu[1], Kui Ren[2], Wenjing Lou[1], and Jin Li[2]

[1] Department of ECE, Worcester Polytechnic Institute, MA 01609
{yscheng@wpi.edu, wjlou@ece.wpi.edu}
[2] Department of ECE, Illinois Institute of Technology, IL 60616
{kren@ece.iit.edu, jin.li@ece.iit.edu}

Abstract. Key-Policy Attribute-Based Encryption (KP-ABE) is a promising cryptographic primitive which enables fine-grained access control over sensitive data. However, key abuse attacks in KP-ABE may impede its wide application especially in copyright-sensitive systems. To defend against this kind of attacks, this paper proposes a novel KP-ABE scheme which is able to disclose any illegal key distributor's ID when key abuse is detected. In our scheme, each bit of user ID is defined as an attribute and the user secret key is associated with his unique ID. The tracing algorithm fulfills its task by tricking the pirate device into decrypting the ciphertext associated with the corresponding bits of his ID. Our proposed scheme has the salient property of *black box* tracing, i.e., it traces back to the illegal key distributor's ID only by observing the pirate device's outputs on certain inputs. In addition, it does not require the pirate device's secret keys to be *well-formed* as is compared to some previous work. Our proposed scheme is provably secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption and the Decisional Linear (DL) assumption.

1 INTRODUCTION

There is a trend that more data are stored or delivered across third parties over Internet for either reliable storage or ease of sharing. For example, individuals would store their personal information on portal web sites such as Google, and commercial content providers may deliver their product through content delivery networks (CDNs) such as Akamai. Such a trend raises the concern that sensitive data stored or cached by these third-party sites will be compromised. Moreover, in some critical or copyright-sensitive application scenarios, it requires differentiated service in the way that, data are defined with sets of attributes and each user is limited to access data of some particular set of attributes or their combinations. In this kind of applications, each user's access privilege is assigned by the user's role or the price that this user paid. One example of this kind of applications is *targeted broadcast* system, e.g., a digital video recorder (DVR) system. In such a system, the content provider might broadcast episodes of TV shows and each of the shows may be assigned a set of attributes such as *name*, *season number*, *genre*, so on and so forth. Users will obtain the access privilege to contents of some particular combination of these attributes by paying the corresponding price to the content provider. The user's access privilege can be encoded as a policy such as ("*name=heros*" AND ("*season 2*" OR "*season 3*")). As content providers might provide their services across third party CDNs, for the purpose of access control it is desirable to encrypt the media products using certain cryptographic primitive since traditional centralized access control methods such as the reference monitor approach might not be suitable in this scenario.

Key-policy attribute-based encryption (KP-ABE) [1] is such a cryptographic primitive that was proposed to resolve the exact problem of fine-grained data access control in one-to-many communications. In KP-ABE, a ciphertext is associated with a set of attributes, and each user secret key is embedded with an access structure which is the logic combination of certain set of attributes. Users can decrypt a ciphertext if and only if the set of attributes associated with the ciphertext satisfy the access structures embedded in their secret keys. Beside this property, KP-ABE also has nice properties of collusion resistance and provable security under standard difficulty assumptions. All these properties seem to make KP-ABE a perfect tool to enforce access control in the above copyright-sensitive applications.

However, the following issue may impede its direct application in targeted broadcast systems of our interests: In the current KP-ABE construction [1], a user secret key is defined over an access structure and does not have the one-to-one correspondence with any particular user. This results in the fact that a paid user is able to "share" his secret key and abuse his access privilege without being identified. More seriously, pirates may take this advantage to make profits by abusing the access privilege. We call this kind of misbehavior by *key abuse attacks*. As a matter of fact, key abuse attacks are extremely harmful for copyright-sensitive application scenarios. Imagine that in a DVR system protected by KP-ABE, key abusers can easily distribute content decryption keys to others by ways such as sending via email. Due to the cost of this is extremely low, it is more destructive than directly distributing the content itself. Therefore, before KP-ABE can be safely applied to aforementioned applications, key abuse attacks should be well addressed. The ideal way for defending against key abuse attacks is to technically prevent illegal users from using others' decryption keys. However, it is difficult to realize since it may require on-line servers to monitor the usage of user decryption keys, or the user secret key to be physically associated with the user. In conventional broadcast encryption, the issue of key abuse is addressed by using a technique called *traitor tracing* which has been well studied by previous works [2–5]. The key idea of traitor tracing is to enable the content provider to trace any suspicious pirate device and thus discover illegal key distributor's identitie(s) and collect evidences of key abuse. Then the content provider can sue the illegal key distributors by presenting these evidences to law authorities. Specifically, the content provider would choose particular types of ciphertexts and trick pirate devices into decrypting them. Success of decryption will provide the evidence of pirating. At a high level view, we can play the same trick in KP-ABE to defend against key abuse attacks. However, underlying techniques adopted by existing traitor tracing systems can not be directly applied to KP-ABE because receivers are represented individually in conventional broadcast encryption while not in KP-ABE. Therefore, it is desirable to propose a novel solution for defending against key abuse attacks in KP-ABE.

1.1 Our Contribution

In this paper, we resolve the issue and provide an abuse free KP-ABE (AFKP-ABE) scheme based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption and the Decision Linear (D-Linear) assumption. AFKP-ABE has properties of partially collusion resistance and black box tracing according to the definition in [5]. In addition, AFKP-ABE is efficient since both the secret key size and the ciphertext size are $\mathcal{O}(logN)$, where N is the total number of users. The main technical challenge of our construction of AFKP-ABE is to realize black box tracing, i.e., tracing the pirate device only by observing its outputs on some inputs. To achieve this goal, one frequently used method is to trick the pirate device into decrypting tracing ciphertexts and success of decryption will provide the evidence of pirating as mentioned before. In the context of KP-ABE, however, this implies that an unsuspected user may not be able to correctly decrypt a tracing ciphertext even if the attributes embedded in the ciphertext satisfy his access structure, and thus has the chance to detect the ongoing tracing activity. A pirate can take advantage of this and collude with other pirates to detect tracing activities.

The main idea of our construction is as follows. Each user is assigned a unique ID which is chosen from the identity space. Then, we define bits of user identities as attributes and embed them in user secret key. We call these attributes by *identity-related attributes* and other attributes by *normal attributes*. Normal (non-tracing) encryption algorithm associates the identity-related attributes to the ciphertext in the way that all the bits of the identity space are set to "don't care". The tracing algorithm just associates the suspicious identity corresponding identity-related attributes to the ciphertext. This turns out that only the user with the suspicious identity is able to correctly decrypt the tracing ciphertext. Note that in this construction the only difference between a normal encryption algorithm and the tracing algorithm is on the input set of identity-related attributes. To make the tracing algorithm indistinguishable from the regular encryption algorithm, we hide these identity-related attributes when encrypting so that pirate devices are not able to tell which and how many identity-related attributes are used. In addition we also hide some of the normal attributes. The intuition behind this is to prevent the pirate device from being able to check if normal attributes of the ciphertext satisfy his access structure and thus detect the tracing activity. We achieve the goal of hiding attributes using the technique from anonymous ciphertext-policy attribute-based encryption [6] in which the ciphertext policy is hidden to receivers. Our definition of the KP-ABE tracing system is based on the definition of the traitor tracing system by Boneh et al. [5].

1.2 Related Work

Attribute-Based Encryption Sahai and Waters [7] first introduced attribute-based enctyption (ABE) for encrypted access control. In an ABE system, both the user' private key and the ciphertext are associated with a set of attributes. If only at least k attributes overlap between the ciphertext and his private key, can the user decrypt the ciphertext. Based on ABE, Goyal et al. [1] proposed a key-policy attribute-based encryption (KP-ABE) scheme and introduced the concept of ciphertext-policy attribute-based encryption (CP-ABE). The first CP-ABE construction was proposed by Bethencourt et al. [8]. Cheung et al. [9] proposed the first provably secure CP-ABE. In CP-ABE, the user secret key is associated with a set of attributes and ciphertexts are embedded with an access structure. A user is able to decrypt the ciphertext only if the attributes associated with his secret keys satisfy the access structure of the ciphertext. KP-ABE is defined in the reverse way than CP-ABE. User secret keys in KP-ABE are embedded with an access structure and ciphertexts are associated with a set of attributes. Successful decryption of the ciphertext requires a match between the user's access structure and the ciphertext attribute set.

Anonymous CP-ABE In conventional CP-ABE schemes [8, 9], ciphertext policies should be revealed in the ciphertext so that receivers are able to combine correct secret keys for decryption. To better protect user privacy, some application scenarios may require ciphertext policies to be hidden to receivers. We call this branch of CP-ABE schemes by *anonymous CP-ABE*. The first anonymous CP-ABE scheme was proposed by Kapadia et al. [10]. However, this scheme is not collusion-resistant and it needs an online semi-trusted server to participate in data encryption. Yu et al. [11] proposed a collusion-resistant anonymous CP-ABE based on [9]. But the security of this scheme is based on a strong assumption (SXDH). Nishide et al. [6] proposed the first provably secure anonymous CP-ABE based on the DBDH assumption and the D-Linear assumption. In their proposed scheme, each attribute could have several values. A public key component is defined over each value of an attribute. User secret key is associated with exactly one value of each attribute. The ciphertext has a *well-formed* ciphertext component for each intended attribute value and *mal-formed* ciphertext components for unintended attribute values. It sets an attribute as "don't care" by presenting well-formed ciphertext components for all the values of this attribute. If there is one ciphertext component corresponding to the user attributes is mal-formed, this user will not be able to decrypt the ciphertext. Because the scheme is designed in such a way that it is hard to distinguish well-formed ciphertext components from mal-formed ones, receivers are not able to tell which or how many attributes appear in the ciphertext policy. Our construction is partially based on this scheme. We refer to [6] for more details on this scheme. Anonymous CP-ABE can also be realized by using a recently invented cryptographic primitive called predicate encryption by Katz et al. [12]. However, their construction requires the bilinear group to be of the order of product of three large primes. Moreover, their security proof is based on new complexity assumptions.

Traitor Tracing Traitor tracing systems were proposed for use in broadcast environments to help content providers trace back to the original source of pirates. In a traitor tracing system, each user (with a decoder) is assigned a personal decryption key. The content provider encrypts the content such that only authorized users are able to decrypt. Suppose a group of colluding users P contribute their personal keys to build a pirate decoder. The tracing scheme should be able to trace back to each member of P. The first traitor tracing system is proposed by Chor et al [2]. Since that, many traitor tracing schemes [3–5] have been proposed. These scheme can be categorized by the following properties [5]: public key/private key broadcast encryption, public/private traceability, collusion resistance, black box tracing, stateful/stateless decoder. For example, [5] is a traitor tracing system for public key broadcast and enables private black box tracing against arbitrary colluding stateless decoders. Other important properties of a traitor tracing system include secret key size and ciphertext size.

The rest of this paper is organized as follows. Section 2 reviews some technique preliminaries pertaining to our construction. Section 3 presents formal definitions and models of our proposed abuse free key-policy attribute-based encryption scheme. In section 4 we give our construction of such a scheme as well as our security proof to it. In section 5, we discuss potential application scenarios in which our scheme would be applicable. We conclude this paper in Section 6.

2 Preliminaries

2.1 Bilinear Maps

Our design is based on some facts about groups with efficiently computable bilinear maps.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$. A bilinear map is is an injective function $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ with the following properties:

1. *Bilinearity*: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. *Non-degeneracy*: $e(g, g) \neq 1$.
3. *Computability*: There is an efficient algorithm to compute $e(u, v) for \forall\, u, v \in \mathbb{G}_0$.

2.2 Complexity Assumptions

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_0$. The DBDH assumption [13] states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuples $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with non-negligible advantage.

The Decision Linear (D-Linear) Assumption Let $z_1, z_2, z_3, z_4, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_0$. The D-Linear assumption [14] states that that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z_3+z_4})$ from the tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z})$ with non-negligible advantage.

3 Definitions and Models

In this section, we present the definition of our abuse-free KP-ABE (AFKP-ABE) scheme as well as its security definition. The security definition of our scheme is consistent to traitor tracing schemes [5].

3.1 Description of AFKP-ABE

The AFKP-ABE scheme has the following five algorithms:

Setup$(1^\lambda$, n) The setup algorithm is a randomized algorithm. It takes as input the security parameter 1^λ and n, the length of a user identity. It outputs a master key MK and public parameters PK.

Enc$(M,\ \gamma, PK)$. The encryption algorithm is a randomized algorithm. It takes as input a message M, a set of attributes γ, and the public parameters PK. It outputs a ciphertext E. On different input γ, this algorithm can be used either for normal (non-tracing) operations of content distribution, or for the purpose of tracing.

KeyGen$(T,\ MK,\ PK)$. The key generation algorithm is a randomized algorithm. It takes as input an access structure T, the master secret key MK, and the public parameters PK. It outputs a user secret key SK.

Dec$(E,\ SK,\ PK)$. The decryption algorithm is a deterministic algorithm. It takes as input the ciphertext E for a set of attributes γ, a user secret key SK for an access structure T, and the public parameters PK. If $\gamma \models T$, i.e., γ satisfies T, it outputs the message M. Otherwise it outputs $\perp$ with overwhelming probability.

Trace$^{\mathcal{D}}(\varepsilon)$ This algorithm takes input a parameter ε (which should be polynomially related to λ), and has black-box access to an ε-useful decoder box $\mathcal{D}$ which is constructed by the adversary. It outputs a set of guilty colluders in polynomial time.

3.2 Security Definition

The security of ABKP-ABE is defined by the following two security games.

Game 1. The first game captures the idea of *Semantic Security.* In our scheme we follow the definition of the standard game used by KP-ABE [1] which proceeds with the following steps.

- *Init* The adversary declares the set of attributes, γ, that he wishes to be challenged upon.
- *Setup* The challenger runs the Setup algorithm of AFKP-ABE and gives the public parameters to the adversary.
- *Phase 1* The adversary is allowed to issue queries for private keys for many access structures T_i , where $\gamma \nvDash T_i$ for all i.
- *Challenge* The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with γ. The ciphertext is passed to the adversary.
- *Phase 2* Phase 1 is repeated.
- *Guess* The adversary outputs a guess b_0 of b.

The advantage of an adversary $\mathcal{A}$ winning this game is defined as $Adv_{SS} = Pr[b_0 = b] - \frac{1}{2}$.

Game 2. The second game captures the notion of *Traceability against partial collusion*. Our definition of the traceability game is based on that of [5]. Given λ, n, and ε, the game proceeds with the following steps.

- *Setup* The adversary $\mathcal{A}$ outputs a set $U = \{u_1, u_2, \ldots, u_t\}$ of colluding users with the only restriction that no pair of users have exactly the same access privilege. The access structure associated with user $u_i \in U$ is denoted by T_i.
- *Key Generation* The challenger runs the key generation algorithm $KeyGen$ to provide the user secret key for each user in U.
- The adversary $\mathcal{A}$ outputs a pirate device $\mathcal{D}$.
- The challenger runs $Trace^{\mathcal{D}}(\varepsilon)$ to obtain a set S.

We say that the adversary $\mathcal{A}$ wins the game if the following two conditions hold:

1. The decoder $\mathcal{D}$ is ε-useful. That is, for a randomly chosen M in the finite message space, we have that $Pr[\mathcal{D}(Enc(M, \gamma, PK)) = M] \geq \varepsilon$ if there exists a user $u_i \in U$ with $\gamma \models T_i$, where γ is chosen in the way that Enc runs for normal (non-tracing) operation.
2. The set S is either empty, or is not a subset of U.

We denote the probability that the adversary $\mathcal{A}$ wins this game by Adv_{TR}. If U contains exactly one user, this game captures the notion of *Traceability against single pirate*.

Definition 1. *We say that AFKP-ABE is secure if Adv_{SS} and Adv_{TR} are negligible (in λ) for any polynomial time adversary $\mathcal{A}$ and any constant $\varepsilon > 0$.*

To prove the security of AFKP-ABE in Game 2, another required security game is the *Indistinguishability Game* which captures the notion that, it is hard to distinguish ciphertexts generated by normal (non-tracing) operations from those generated by tracing operations. Its concrete security definition is given in Appendix B.

4 Our Construction

In this section, we present our construction of the secure AFKP-ABE scheme.

4.1 Main Idea

The intuition of our construction can be summarized as the follows. We define a n-bit user identity space and each bit of them is defined as an attribute with two occurrences, one for bit value 0 and the other for bit value 1. Each user is then assigned a unique ID from the identity space. The encryption algorithm will associate these identity-related attributes to the ciphertext in the following way: for normal (non-tracing) operations, all these n attributes are set as "don't care"; for tracing operations, they are set to represent the suspicious identity. In tracing operations, a user is able to decrypt the ciphertext only if his identity equals the suspicious one. To make tracing ciphertexts indistinguishable from normal ciphertexts, we hide these identity-related attributes in the way that any user is not able to tell which and how many of them are set as "interested" (i.e., not "don't care"). In addition, we also hide some normal attributes so that upon a fail decryption the user can not tell if it is caused by the mismatch of his ID or by his access privilege (without considering his ID). Thus, he is not able to distinguish a tracing activity from a normal (non-tracing) one. The security goal of our construction is to build such a KP-ABE scheme in which 1) any user without the correct decryption key is not able to tell a single bit of the message, and 2) given a pirate device, the authority is able to trick it into decrypting tracing ciphertexts and thus discover the identity of the original owner of the decryption key held by this device.

Definition of Attributes We define three set of attributes: *public normal attributes*, *hidden normal attributes* and *hidden identity-related attributes*. We denote the universe of each of them by $\mathcal{U}_{PN}$,$\mathcal{U}_{HN}$, and $\mathcal{U}_{HID}$ respectively. The letter P in the subscription denotes the word "public", H means "hidden", N represents "normal", and ID is the abbreviation of "identity". $\mathcal{U}_{PN}$ and $\mathcal{U}_{HN}$ contain attributes to be used by normal encryptions. $\mathcal{U}_{HID}$ contains identity-related attributes for describing the suspected user's identity and is particularly used for tracing. In ciphertexts, the associated attributes from $\mathcal{U}_{HN}$ and $\mathcal{U}_{HID}$ have to be hidden such that any receiver is not able to tell which and how many of them are used, while attributes from $\mathcal{U}_{PN}$ are public. Each attribute in $\mathcal{U}_{HID}$ has two occurrences, one for bit value 0 and the other for bit value 1. Similarly, we assume that attributes in $\mathcal{U}_{HN}$ also have binary values like those in $\mathcal{U}_{HID}$. This assumption is just for concise presentation of our scheme. Extending our scheme to support the non-binary case is trivial. From now on we will call the union of $\mathcal{U}_{HID}$ and $\mathcal{U}_{HN}$ as *hidden attributes* by capturing their common property of "hidden". We denote the universe of hidden attributes as $\mathcal{U}_H$, and thus $\mathcal{U}_H = \mathcal{U}_{HN} \cup \mathcal{U}_{HID}$. We denote the number of attributes in $\mathcal{U}_{HN}$ by m and that in $\mathcal{U}_{PN}$ by k. Therefore, the total number of hidden attributes is $m + n$.

According to the above discussion, it is clear that in a ciphertext there could be three types of attributes: attributes from $\mathcal{U}_{PN}$, attributes from $\mathcal{U}_{HN}$,

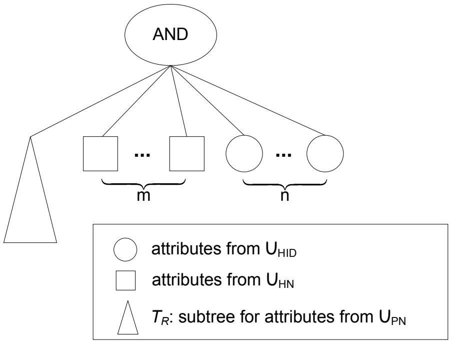
AND
...
...
m
n
attributes from U_{HID}
attributes from U_{HN}
T_R: subtree for attributes from U_{PN}

Fig. 1. Illustration of the construction of our access structure

and those from $\mathcal{U}_{HID}$. We denote the set of these three type of attributes in a ciphertext by γ_{PN}, γ_{HN}, and γ_{HID} respectively. Therefore, we have $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$, where γ represents the set of all the attributes interested by the encryptor.

Access Structure Our definition of the access structure (implemented using an access tree) is the same as KP-ABE [1], i.e., each interior node of the tree is a threshold gate and the leaves are associated with attributes. However, our construction has the following restrictions on the access structure: (1) each access structure should deal with all the hidden attributes and all of them should appear on the second layer of the tree; (2) the root node has to be an AND gate; (3) all the attributes from $\mathcal{U}_{PN}$ should appear in a subtree which we denote by T_R. Interior nodes of the subtree T_R could be any kind of threshold gates. The structure of the access tree in our construction is illustrated by Fig. 1. In addition, each non-root node has a unique index given by its parent. For the convenience of representation, we will denote a node x's parent by x_{pa} and x's index by $idx(x)$.

4.2 AFKP-ABE Scheme

In the description, $\mathbb{G}_0$ is a bilinear group of prime order p and g is a generator of $\mathbb{G}_0$. We use $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ to represent a bilinear map. The Lagrange coefficient $\Delta_{i,S}(x)$ is defined as follows, where $i \in \mathbb{Z}_p$, $x \in \mathbb{Z}_p$ are variables, and $S \subset \mathbb{Z}_p$ is some set.

$$\Delta_{i,S}(x) := \prod_{j \in S \setminus \{i\}} \frac{x - j}{i - j}.$$

We use strings of length n to represent user IDs. “don’t care” bit of an ID is represented by a “$*$”.

Setup$(1^\lambda, n)$ Define $\mathcal{U}_H = \{1, \cdots n, n+1, \cdots m+n\}$, where the first n elements are for $\mathcal{U}_{HID}$ and the last m for $\mathcal{U}_{HN}$, and $\mathcal{U}_{PN} = \{1, 2, \cdots k\}$. For each attribute

$i \in \mathcal{U}_{PN}$, choose a random number t_i from $\mathbb{Z}_p$. Then for each hidden attribute $j \in \mathcal{U}_H$, choose random numbers $\{a_{j,t}, b_{j,t}\}_{t=0,1}$ from $\mathbb{Z}_p$ and random points $\{A_{j,t}\}_{t=0,1}$ from $\mathbb{G}_0$. Finally, choose a random number y from $\mathbb{Z}_p$. The public parameters PK are published as

$$PK = (Y = e(g,g)^y, \{T_i = g^{t_i}\}_{i \in \mathcal{U}_{PN}}, \{A_{j,t}^{a_{j,t}}, A_{j,t}^{b_{j,t}}\}_{j \in \mathcal{U}_H, t=0,1})$$

and the master key MK is

$$MK = (y, \{t_i\}_{i \in \mathcal{U}_{PN}}, \{a_{j,t}, b_{j,t}\}_{j \in \mathcal{U}_H, t=0,1})$$

Enc(M, γ, PK) Define $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ as mentioned before. Let the ID represented by γ_{HID} be $X_n X_{n-1} \cdots X_1$, where $X_i = 0, 1$ or $*$ for each $1 \leq i \leq n$. The encryptor generates ciphertext components for γ_{HID} as follows. First choose a random number s from $\mathbb{Z}_p$. Then for each $1 \leq i \leq n$, pick random numbers $r_{i,0}$ and $r_{i,1}$ from $\mathbb{Z}_p$, and compute tuples $\{[\hat{E}_{i,t}, \check{E}_{i,t}]\}_{t=0,1}$ as follows.

(1) If $X_i = b$, where $b = 0|1$, the encryptor sets $[\hat{E}_{i,1-b}, \check{E}_{i,1-b}]$ as random (*mal-formed*), and $[\hat{E}_{i,b}, \check{E}_{i,b}] = [(A_{i,b}^{b_{i,b}})^{r_{i,b}}, (A_{i,b}^{a_{i,b}})^{s-r_{i,b}}]$ (*well-formed*).

(2) If $X_i = *$, for $t = 0, 1$ the encryptor sets $[\hat{E}_{i,t}, \check{E}_{i,t}] = [(A_{i,t}^{b_{i,t}})^{r_{i,t}}, (A_{i,t}^{a_{i,t}})^{s-r_{i,t}}]$ (*well-formed*).

Ciphertext components for γ_{HN} are generated in the same way as γ_{HID}. The encryptor generates ciphertext components for γ_{PN} as follows. For each $i \in \gamma_{PN}$, compute $E_i = T_i^s$. Finally, the ciphertext is output as follows.

$$E = (\gamma_{PN}, \tilde{E} = MY^s, E_0 = g^s, \{E_i\}_{i \in \gamma_{PN}}, \{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{j \in \gamma_{HN} + \gamma_{HID}})$$

KeyGen$(T,\ MK,\ PK)$ The access structure T is defined as mentioned before: the root node of the tree is an AND gate, all the hidden attributes appear on the second layer of the tree, and all the public normal attributes are in the subtree T_R. The trusted authority generates the user secret key as follows.

(1) For the subtree T_R, choose a polynomial q_x for each node x, including all the leaf nodes, of the tree in the top-down manner as follows. Starting from the root node r of T_R (with the threshold value k_r), choose a random number u from $\mathbb{Z}_p$ and set $q_r(0) = u$. Then randomly choose $k_r - 1$ other points to define the $(k_r - 1)$-degree polynomial q_r completely. For any other node x, q_x is generated in the same way and $q_x(0) = q_{x_{pa}}(idx(x))$.

After having defined the polynomials, the following secret key component is generated for each leaf node x in T_R:

$$D_x = g^{\frac{qx(0)}{t_i}}$$

where i denotes the attribute in $\mathcal{U}_{PN}$ associated with node x. We use $\mathcal{L}_{TR}$ to represent the set of all the leaf nodes in T_R.

(2) Secret key components for attributes from $\mathcal{U}_{HID}$ are generated as follows. Assume the user is assigned a unique identity $ID = X_n X_{n-1} \cdots X_1$, where $X_i = 0|1$ for each $1 \leq i \leq n$. Then for each attribute i in $\mathcal{U}_{HID}$, the authority

chooses random numbers v_i and λ_i from $\mathbb{Z}_p$ and outputs a triple $[\tilde{D}_i, \hat{D}_i, \check{D}_i]$ as follows.

$$\tilde{D}_i = g^{v_i}(A_{i,X_i})^{a_{i,X_i} b_{i,X_i} \lambda_i}, \quad \hat{D}_i = g^{a_{i,X_i}\lambda_i}, \quad \check{D}_i = g^{b_{i,X_i}\lambda_i}.$$

(3) Secret key components for attributes from $\mathcal{U}_{HN}$ are generated in the same way as $\mathcal{U}_{HID}$.

(4) The authority sets $v = \sum_{i \in \mathcal{U}_H} v_i$ and generates a secret key component $D_0 = g^{y-u-v}$.

Finally, the authority outputs the following as the user secret key (SK):

$$SK = (D_0, \{D_i\}_{i \in \mathcal{L}_{TR}}, \{\tilde{D}_i, \hat{D}_i, \check{D}_i\}_{i \in \mathcal{U}_H})$$

Dec(E, SK, PK) The receiver decrypts the ciphertext E by applying his secret key components to the ciphertext as follows.

(1) Apply secret key components for public normal attributes to the ciphertext. For each leaf node x of T_R, assuming x is associated with attribute $i \in \mathcal{U}_{PN}$, calculate the following (the result is denoted by F_x):

$$F_x = \begin{cases} e(D_i, E_i) = e(g,g)^{sq_x(0)}, \text{ if } x \in \gamma_{PN} \ ; \\ \perp, \qquad\qquad\qquad\qquad\quad \text{otherwise.} \end{cases} \tag{1}$$

Then execute recursively for each non-leaf node z of T_R in the bottom-up manner as follows. For each child node x of z, if $F_x \neq \perp$ add x into a set S_z until S_z has k_z elements, where the set S_z is initialized to empty. If not able to construct such a k_z-sized set S_z, let $F_z = \perp$. Otherwise, calculate F_z as follows.

$$\begin{aligned} F_z &= \prod_{x \in S_z} F_x^{\Delta_{x,S_z}(0)} \\ &= \prod_{x \in S_z} (e(g,g)^{sq_x(0)})^{\Delta_{x,S_z}(0)} \\ &= e(g,g)^{sq_z(0)} \end{aligned}$$

where derivation of the last two steps holds because $q_x(0) = q_z(idx(x))$ and $q_z(0) = \sum_{x \in S_z}(q_z(idx(x)) \cdot \Delta_{x,S_z}(0))$.

This recursion ends up with outputting $F_r = e(g,g)^{sq_r(0)}$ if $\gamma_{PN} \models T_R$. Since $q_r(0) = u$, we have $F_r = e(g,g)^{su}$.

(2) Apply secret key components for hidden attributes to the ciphertext. If the set of hidden attributes in the access structure contains all the attributes in γ_{HN} and γ_{HID}, output the result F_H as follows.

$$\begin{aligned} F_H &= \prod_{i \in \mathcal{U}_H} \frac{e(E_0, \tilde{D}_i)}{e(\hat{E}_i, \hat{D}_i) e(\check{E}_i, \check{D}_i)} \\ &= e(g,g)^{sv} \end{aligned}$$

The message can be output as follows

$$
\begin{aligned}
M &= \frac{\tilde{E}}{e(E_0, D_0)F_r F_H} \\
&= \frac{Me(g,g)^{ys}}{e(g^s, g^{y-u-v})e(g,g)^{su}e(g,g)^{sv}}
\end{aligned}
$$

Trace$^{\mathcal{D}}(\varepsilon)$ This algorithm takes as input ε and a ε-useful pirate device $\mathcal{D}$. We first show how to trace $\mathcal{D}$ which just holds one decryption key as follows. The tracing algorithm repeats the following steps $\frac{1}{\varepsilon}$ times for each identity ID_i in the system identity list:

– Step 1. Choose a set of attributes $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ such that γ satisfies the access structure of ID_i and γ_{HID} just contains the attributes corresponding to bits of ID_i.
– Step 2. Choose a random message M from the finite message space. Let $E \leftarrow Enc(M, \gamma, PK)$.
– Step 3. Test if $\mathcal{D}$ correctly decrypts E. If it does, stop and return with ID_i. Otherwise continue.

If at the end of these repetitions the algorithm does not return with any identity, return FAIL and stop the experiment. Tracing $\mathcal{D}$ which holds more than one decryption keys is similar with the exception that, in step 3 add ID_i into the guilty user set S instead of returning immediately, where S is initialized as empty. If at the end of these repetitions S is empty, return FAIL and stop the experiment.

4.3 Security Proof

We show the security of our scheme as follows.

Lemma 41 *If a polynomial-time adversary $\mathcal{A}$ can win Game 1 with non-negligible advantage Adv_{SS}, then we can build a simulator $\mathcal{B}$ that is able to solve the DBDH problem with advantage $\frac{1}{2}Adv_{SS}$.*

Proof. Security proof of this Lemma is presented in Appendix A.

Lemma 42 *If a polynomial-time adversary $\mathcal{A}$ can win our Indistinguishability Game (see Appendix B) with advantage Adv_{IND}, then we can build a simulator $\mathcal{B}$ that is able to solve the D-Linear problem with advantage $\frac{1}{2}Adv_{IND}$.*

Proof. Security proof of this Lemma is presented in Appendix B.

Lemma 43 *If Adv_{IND} and Adv_{SS} are negligible, Adv_{TR} is negligible.*

Proof. Given a pirate device $\mathcal{D}$, our tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\varepsilon)$ will try with each identity ID_i in the system identity list. We denote the attribute set chosen for testing ID_i by $\gamma^i = \gamma^i_{PN} \cup \gamma^i_{HN} \cup \gamma^i_{HID}$. We define the corresponding attribute set used for normal (non-tracing) encryption as $\bar{\gamma}^i = \gamma^i_{PN} \cup \gamma^i_{HN} \cup \bar{\gamma}^i_{HID}$. The only difference between the two sets of attributes is that, in γ^i_{HID} all the attributes corresponding to bits of ID_i are set as "interested", but in $\bar{\gamma}^i_{HID}$ all the identity-related attributes are set as "don't care". Based on this definition, we define the following two probabilities:

$$p_i = \Pr[\mathcal{D}(Enc(M, \gamma^i, PK)) = M]$$
$$p = \Pr[\mathcal{D}(Enc(M, \bar{\gamma}^i, PK)) = M]$$

where M is a random message picked from the message space. We distinguish between the following three types of ε-useful pirate devices that the pirate can generate, where ε is some fixed constant:

1. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is non-negligible for some identity ID_i.
2. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is negligible for each identity ID_i, but the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\varepsilon)$ outputs an empty set.
3. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is negligible for each identity ID_i, but the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\varepsilon)$ outputs a set which is not contained in the set of colluding users U.

It is obvious that we can use any pirate producing type 1) devices to win the *Indistinguishability Game* with non-negligible advantage. We now show the rough idea of how we can use any pirate producing type 2) devices to win the *Indistinguishability Game* with non-negligible advantage. Assume the set of colluding users that the pirate claims to be able to collect is $U = \{u_1, u_2, \cdots, u_t\}$. Now denote the challenger of the *Indistinguishability Game* is $\mathcal{C}$, the simulator we want to build is $\mathcal{B}$, and the pirate is $\mathcal{A}$. Then the simulator we build executes as follows.

- Init. $\mathcal{B}$ presents $\mathcal{C}$ two attribute sets $\gamma_0 = \gamma^i$ and $\gamma_1 = \bar{\gamma}^i$ to be challenged upon, where γ^i is the attribute set that can be used to test user $u_i \in U$ by our tracing algorithm.
- Setup. $\mathcal{C}$ generates public parameters and give them to $\mathcal{B}$.
- Phase 1. $\mathcal{B}$ asks $\mathcal{C}$ to give him secret keys for all the users in U. Then $\mathcal{B}$ gives all these keys to $\mathcal{A}$ to answer key queries in the key generation phase of Game 2.
- Challenge. $\mathcal{B}$ submits two equal length messages M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ flips a coin and encrypts M_b with γ_b. Then the ciphertext is given to $\mathcal{B}$.
- Phase 2. $\mathcal{B}$ submits more secret key queries.
- Guess. $\mathcal{B}$ asks $\mathcal{A}$ to decrypt the ciphertext given by $\mathcal{C}$. If the message returned by $\mathcal{A}$ is one of M_1 and M_0, $\mathcal{B}$ answers $b_0 = 1$. Otherwise, $\mathcal{B}$ answers $b_0 = 0$.

The advantage for our simulator $\mathcal{B}$ to win the *Indistinguishability Game* is $\frac{1}{\varepsilon}$ times the advantage that the type 2) devices, which are generated by $\mathcal{A}$, output the empty set.

It is easy to show that type 3) devices can be used to win Game 1 (the semantic security game). The intuition is that, type 3) devices can correctly decrypt a message which is encrypted for users whose secret keys are not known to type 3) devices with non-negligible advantage.

4.4 Efficiency Analysis

In AFKP-ABE, both the ciphertext size and the secret key size are linear to n, where n is the number of bits in the identity space. As the maximum number of users it can represent is $N = 2^n$, the complexity can be written as $\mathcal{O}(logN)$, where N is the total number of users. To trace a pirate, AFKP-ABE needs to try with every user's identity in the system list. When the number of users in a system is large, the tracing algorithm would be inefficient. To resolve this issue, we can first test with some normal ciphertexts using combinations of normal attributes. For example, we can use different combinations of attributes like location, age, etc. In practice, this process will hopefully rule out a significant portion of users. Our tracing algorithm can just test over the remaining set of users.

5 Application Scenarios of Our Scheme

In general, our proposed scheme is applicable to systems where 1) data can be categorized by their attributes and a user access privilege should be defined in the way that just allows the user to access certain intended subset of resources; 2) abuse of the access privilege should be prohibited. As we mentioned before, one important application scenario of our abuse free KP-ABE scheme is the area of copyright-sensitive targeted broadcast, especially commercial media broadcast systems. In these systems, contents usually have their commercial values and abuse of the access privilege usually cause legal concerns. Another important application scenario of our proposed scheme would be audit log systems. As these systems would be used widely used in applications such as network management, audit logs may contain sensitive information and disclose of them to unauthorized parties would cause security concerns or privacy violations. Recently, we also witnessed application of KP-ABE in wireless networks environment. In [15], Yu et al. proposed a fine-grained data access control scheme for wireless sensor networks for mission-critical applications. In this paper, data access control is well resolved by combining KP-ABE and some other cryptographic primitives. However, the issue of access privilege abuse is not addressed since it is yet another serious issue if we consider the application of mission-critical scenarios such as battle fields. We believe our AFKP-ABE can serve to enhance their proposed scheme as the complexity of AFKP-ABE in terms of ciphertext size and secret key size is just $\mathcal{O}(logN)$, where N is the total number of users.

6 Conclusion and Future Work

In this paper, we focus on the key abuse attacks in KP-ABE enabled broadcast systems and proposed an abuse free KP-ABE (AFKP-ABE) scheme. To defend against the key abuse attacks, we introduce hidden attributes in the system such that the tracing algorithm can use them to identify any single pirate or partial colluding users. Our design enables black boxing tracing and does not require the well-formness of the user secret key. The complexity of AFKP-ABE in terms of ciphertext size and user secret keys size is just $\mathcal{O}(logN)$, where N is the total number of users. Our scheme is provably secure under DBDH assumption and D-Linear assumption. As a future work, we may focus on designing a tracing system against arbitrary colluders.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0626601, CNS-0746977, CNS-0831628, and CNS-0831963.

References

1. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS '06*, 2006, pp. 89–98.
2. B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *CRYPTO'94*. London, UK: Springer-Verlag, 1994, pp. 257–270.
3. D. Boneh and M. K. Franklin, "An efficient public key traitor tracing scheme," in *CRYPTO'99*. London, UK: Springer-Verlag, 1999, pp. 338–353.
4. A. Kiayias and M. Yung, "Traitor tracing with constant transmission rate," in *EUROCRYPT'02*. London, UK: Springer-Verlag, 2002, pp. 450–465.
5. D. Boneh, A. Sahai, and BrentWaters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT'06*. London, UK: Springer-Verlag, 2006.
6. T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *ACNS'08*. LNCS 5037, 2008, pp. 111–129.
7. A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT'05*, ser. Lecture Notes in Computer Science, R. Cramer, Ed., vol. 3494. Springer, 2005, pp. 457–473.
8. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *SP'07*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 321–334.
9. L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *CCS'07*. New York, NY, USA: ACM, 2007, pp. 456–465.
10. A. Kapadia, P. Tsang, and S. Smith, "Attribute-based publishing with hidden credentials and hidden policies," in *NDSS'07*. LNCS 5037, 2007, pp. 179–192.
11. S. Yu, K. Ren, and W. Lou, "Attribute-based content distribution with hidden policy," in *NPSEC'08*, 2008.

12. J. Katz, A. Sahai, and B.Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in *Eurocrypt’08*. LNCS 4965, 2008, pp. 146–162.
13. D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *CRYPTO’01*. LNCS 2139, 2001, pp. 213–229.
14. D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” in *CRYPTO’04*. LNCS 3152, 2004, pp. 41–55.
15. S. Yu, K. Ren, and W. Lou, “Fdac: Toward fine-grained distributed data access control in wireless sensor networks,” in *IEEE INFOCOM’09*, 2009.

Appendix

A. Security of Lemma 41

Proof. In the DBDH game, the challenger chooses random numbers a, b, c from $\mathbb{Z}_p$ and flips a fair coin μ. If $\mu = 0$, set $z = abc$; If $\mu = 1$, set z as a random value in $\mathbb{Z}_p$. $\mathcal{B}$ is given (A, B, C, Z)=$(g^a, g^b, g^c, e(g, g)^z)$ and asked to output μ. To answer this challenge, $\mathcal{B}$ then simulates Game 1 as follows.

Init $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the set of attributes $\gamma = \gamma_{PN} + \gamma_{HN} + \gamma_{HID}$ it wants to be challenged upon. We denote the identity represented by γ_{HID} by $X_n X_{n-1} \cdots X_0$, where $X_i = 0, 1$ or $*$, for $1 \leq i \leq n$.We denote the set $\gamma_{HN} + \gamma_{HID}$ by γ_H.

Setup $\mathcal{B}$ creates public parameters as follows. First, set $Y = e(A, B) = e(g, g)^{ab}$. Then, for each attribute $i \in \mathcal{U}_{PN}$, generate T_i by the following steps:

- choose a random number $t_i \in \mathbb{Z}_p$.
- if $i \in \gamma_{PN}$, sets $T_i = g^{t_i}$; otherwise, set $T_i = g^{bt_i} = B^{t_i}$.

For each attribute $i \in \mathcal{U}_{HID}$, choose two random numbers $h_{i,0}$ and $h_{i,1}$ from $\mathbb{Z}_p$. Then proceed as follows.

- if $X_i = *$, $A_{i,t} = g^{h_{i,t}}$, $t = 0, 1$; otherwise, $A_{i,X_i} = g^{h_{i,X_i}}$ and $A_{i,1-X_i} = g^{bh_{i,1-X_i}} = B^{h_{i,1-X_i}}$.
- choose random numbers $\{a_{i,t}, b_{i,t}\}_{t=0,1}$ from $\mathbb{Z}_p$.

Attributes in $\mathcal{U}_{HN}$ are processed in the same way as $\mathcal{U}_{HID}$. Finally, output PK as in the real scheme.

Phase I $\mathcal{A}$ submits a query for secret key of access structure T, where $\gamma \nvDash T$. Note that T has the structure of 1. $\mathcal{B}$ differentiates the following two cases and answers the query accordingly:

Case 1: In this case, $\gamma_{PN} \nvDash T_R$. $\mathcal{B}$ generates secret key components for hidden attributes as in the real scheme. To generate secret key components for attributes attached to T_R, $\mathcal{B}$ defines a recursive function $PolyDef(x)$ and runs it over the root node r of T_R. For each node x in T_R, use k_x and p_x to represent the node's threshold value and the number of its satisfied children respectively (the satisfied child is a child node of x that returns true over γ_{PN}).

PolyDef(x): It is defined by the following steps:

- Define q_x as follows.
  - If x is not r, set $q_x(0) = q_{x_{pa}}(idx(x))$; otherwise, set $q_x(0) = ab + br'$, r' is randomly chosen from $\mathbb{Z}_p$.
  - Select d $(= k_x - 1)$ children of x. For each selected child i, choose a random number r'_i from $\mathbb{Z}_p$ and let $q_x(idx(i)) = br'_i$. This completes the construction of polynomial q_x. Note that, if $p_x \leq d$, the set of selected children should include all the p_x satisfied ones; otherwise, all the d selected children should be satisfied ones. We denote the set of these selected children of x plus x itself by X_s.
- For each remaining child j (not selected by the above step), calculate $q_x(j) = \sum_{i \in X_s} q_x(idx(i)) \varDelta_{i,S_x}(j)$.
- For each child i of x, run $PolyDef(i)$.

When $PolyDef(r)$ terminates, $\mathcal{B}$ completes the construction of the polynomials for all the nodes in T_R. In particular, $p_r(0) = ab + br'$. Note that, in our construction of polynomials, for each node x, the polynomial values have the following properties:

(1) If $q_x(0)$ has the form of $R_x b$, then for each of its children i, $q_i(0)$ $(= q_x(idx(i)))$ has the form of $R_i b$.

(2) If $q_x(0)$ has the form of $C_x ab + R_x b$, then for each of its children i, (i) if $i \in X_s$ (selected), $q_i(0)$ has the form of $R_i b$; otherwise, (ii) $q_i(0)$ has the form of $C_i ab + R_i b$.

(3) In (1) and (2), C_x, R_x, C_i, and R_i are functions of Lagrange coefficients and random numbers (i.e., r'_j's), and independent of a and b.

From these properties, we may categorize a leaf nodes x into one of the following three types:

(1) Type A: $x \in \gamma_{PN}$, i.e., x is a satisfied node. $q_x(0)$ has the form of $R_x b$.

(2) Type B: $x \notin \gamma_{PN}$ but one of x's ancestors (including x itself) is selected by its parent. $q_x(0)$ has the form of $R_x b$.

(3) Type C: all the other leaf nodes, $q_x(0)$ has the form of $C_x ab + R_x b$.

Therefore, the secret key component corresponding to each leaf node x of T_R is given as follows

$$D_x = \begin{cases} g^{\frac{Rx\,b}{tx}} = B^{\frac{Rx}{tx}}, & x \text{ in Type A.} \\ g^{\frac{Rx\,b}{tx\,b}} = g^{\frac{Rx}{tx}}, & x \text{ in Type B.} \\ g^{\frac{Cx\,ab+Rx\,b}{tx\,b}} = A^{\frac{Cx}{tx}} g^{\frac{Rx}{tx}}, & x \text{ in Type C.} \end{cases} \tag{2}$$

The secret key component D_0 of SK is output as follows

$$g^{y-u-v} = g^{ab-q_r(0)-v} = g^{-br'} g^{-v} = B^{-r'} g^{-v}$$

where v is generated when constructing secret key components for hidden attributes. All the other components are generated as in the real scheme.

Case 2: In this case, $\gamma_{PN} \models T_R$, but the hidden attributes of T do not match with γ_H. Let a hidden attribute j that is not intended by γ_{HID} be the witness. $\mathcal{B}$ generates secret key components corresponding to T_R as in the real scheme. $\mathcal{B}$ generates secret key components for hidden attributes as follows.

- For hidden attributes $1 \leq i \leq m+n$, pick v'_i randomly from $\mathbb{Z}_p$. Set $v_j = ab + v'_j$ and $v_i = v'_i$ for every $i \neq j$. Finally set $v = \sum_{i=1}^{m+n} v_i = ab + \sum_{i=1}^{m+n} v'_i$.
- compute the secret key components $[\tilde{D}_j, \hat{D}_j, \check{D}_j]$ of attribute j as follows.

$$\begin{aligned}\tilde{D}_j &= g^{v_j}(A_{j,X_j})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{ab+v'_j}(A_{j,X_j})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{ab+v'_j}(g^{bh_{j,X_j}})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{v'_j}(g^{bh_{j,X_j}})^{a_{j,X_j} b_{j,X_j} \lambda'_j}\end{aligned}$$

  where λ'_j is chosen by $\mathcal{B}$ and $\lambda_j = \frac{a}{h_{j,X_j} a_{j,X_j} b_{j,X_j}} + \lambda'_j$. $\mathcal{B}$ calculates $[\hat{D}_j, \check{D}_j]$ and $[\tilde{D}_i, \hat{D}_i, \check{D}_i]$ for $i \neq j$ as in the real scheme.
- Output D_0 of SK as: $D_0 = g^{ab-u-v} = g^{-u-\sum_{i=1}^{m+n} v'_i}$, where u is generated when constructing secret key components for T_R.

All the other components are generated as in the real scheme.

From the above description, we can see that $\mathcal{B}$ is able to construct a secret key of T in both cases. Furthermore, the distribution of the secret key of T is the same as that in the original scheme. The adversary $\mathcal{A}$ can repeat this step for polynomial times.

Challenge The adversary $\mathcal{A}$ submits two equal length challenge messages m_0 and m_1 to $\mathcal{B}$. $\mathcal{B}$ flips a fair binary coin v and picks out m_v. The ciphertext of m_v is output as: $E = (\gamma_{PN}, \tilde{E} = m_v Z, E_0 = C, \{E_i = C^{t_i}\}_{i \in \gamma_{PN}}, \{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{i \in \gamma_H})$. Note that $\mathcal{B}$ can construct $\{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{i \in \gamma_H}$ because if the occurrence t of attribute i is in γ_H, $A_{i,t}$ does not contain the unknown value b, and if the occurrence t of i is not in γ_H, $\{\hat{E}_{i,t}, \check{E}_{i,t}\}$ are just chosen at random. If $\mu = 0$ it is easy to show that the ciphertext is a valid random encryption of message m_v. Otherwise, if $\mu = 1$, then $Z = e(g,g)^z$ and $\tilde{E} = m_v e(g,g)^z$. Since z is random, $\tilde{E}$ is just a random element of $\mathbb{G}_1$ from the adversary's view and contains no information about m_v.

Phase II The simulator acts exactly as it did in Phase I.

Guess The adversary $\mathcal{A}$ submits a guess v' of v. If $v' = v$, $\mathcal{B}$ outputs $\mu' = 0$, indicating that the given DBDH-tuple is a valid one. Otherwise it outputs $\mu' = 1$, indicating that the given DBDH-tuple is just a random quadruple. In the case of $\mu = 1$, the ciphertext E contains no information about m_v. Therefore, v' is just a random guess of v, and thus μ' is just a random guess of μ. Thus, we have $Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, the ciphertext E is a valid encryption of m_v. Since by definition $\mathcal{A}$ has the advantage of Adv_{SS} to output a correct guess, i.e., $v' = v$, $\mathcal{B}$ outputs $\mu' = 0$ with the probability of $\frac{1}{2} + Adv_{SS}$, i.e., $Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + Adv_{SS}$. Therefore, the overall advantage of $\mathcal{B}$ in the DBDH game is $\frac{1}{2}Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + Adv_{SS}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2}Adv_{SS}$.

6.1 Indistinguishability Game

This game captures the idea that ciphetexts generated by tracing operations are indistinguishable from those generated by normal (non-tracing) operations.

In AFKP-ABE, these two types of ciphertexts are generated by running our encryption algorithm over different sets of attributes. To differentiate these two types of ciphertexts is actually equal to telling which set of attributes are used in a given data encryption operation. As we discussed, the attribute set γ used for an encryption operation is composed of three disjunctive subsets, i.e., $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$. In a tracing operation, we set γ_{HID} to represent the suspicious identity, while in a normal (non-tracing) operation we set γ_{HID} to represent the identity of "$* * \cdots *$", i.e., each bit if ID is set as "don't care". We define the *Indistinguishability Game* by the following steps:

Init The adversary $\mathcal{A}$ selects two sets of attributes to be challenged upon: $\gamma_0 = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ and $\gamma_1 = \gamma_{PN} \cup \gamma_{HN} \cup \gamma^*_{HID}$, where γ_{HID} represents a certain identity ID_i, and γ^*_{HID} denotes the identity of "$* * \cdots *$", i.e., each bit if ID is set as "don't care". $\mathcal{A}$ submits these two sets of attributes to the challenger $\mathcal{C}$.

Setup The challenger $\mathcal{B}$ runs the setup algorithm of AFKP-ABE and give public parameters PK to $\mathcal{A}$.

Phase 1 $\mathcal{A}$ asks for the secret key of access structure T. If $(\gamma_0 \models T \wedge \gamma_1 \models T)$ or $(\gamma_0 \not\models T \wedge \gamma_1 \not\models T)$, the challenger $\mathcal{B}$ answers the query and gives $\mathcal{A}$ the corresponding secret key SK_T. The adversary $\mathcal{A}$ can repeat this step polynomially many times.

Challenge $\mathcal{A}$ submits two equal length messages M_0 and M_1 to $\mathcal{B}$. If $\mathcal{A}$ a secret key SK_T for which $(\gamma_0 \models T \wedge \gamma_1 \models T)$, it is required that $M_0 = M_1$. $\mathcal{B}$ flips a binary fair coin b and encrypts M_b using attribute set γ_b. The ciphertext is given to $\mathcal{A}$.

Phase 2 Repeat Phase 1. If $M_0 \neq M_1$, $\mathcal{A}$ can not submit secret key query for access structure T for which $(\gamma_0 \models T \wedge \gamma_1 \models T)$.

Guess The adversary $\mathcal{A}$ outputs a guess b' of b.

Proof. We use a series of games to prove the security of this game as [6]. *Game Ind*$_1$ is defined in the same way as the original game except that in γ_0, γ_{HID} represents the identity of "$* * \cdots * X_1$", i.e., the upper $n-1$ bits are set as "don't care" but keep the first bit the same as in the original game. *Game Ind*$_2$ is defined in the same way that γ_{HID} represents the identity of "$* * \cdots * X_2 X_1$", i.e., the upper $n-2$ bits are set as "don't care" but keep the first bit the same as in the original game, so on and so forth. Our original game is thus *Game Ind*$_n$. To prove the security of our scheme, it is enough to prove that it is indistinguishable between *Game Ind*$_i$ and *Game Ind*$_{i+1}$. We can use the similar technique used by [6] to prove this.